

Exercise: 6

Name: Pekka Lehtola

How many tasks did you do: 8

Were the tasks easy, ok, difficult: Easy

Do you need help/comments in any task (if yes, to which ones):

1. Explain the following terms:

a. Super class

Super luokalla tarkoitetaan sitä luokkaa mistä ollaan peritty, käytetään myös nimitystä parent

b. Sub class

Sub class tarkoittaa luokkaa mikä on peritty Super classista.

c. Base class

Oman ymmärryksen mukaan, Base class = Super class

d. Derived class

Derived class = Sub class

e. "Is a" relationship

Tällä tarkoitetaan suhdetta luokkien välillä.

Dog **is a** domestic_animal. Domestic_animal **is a** Mammal

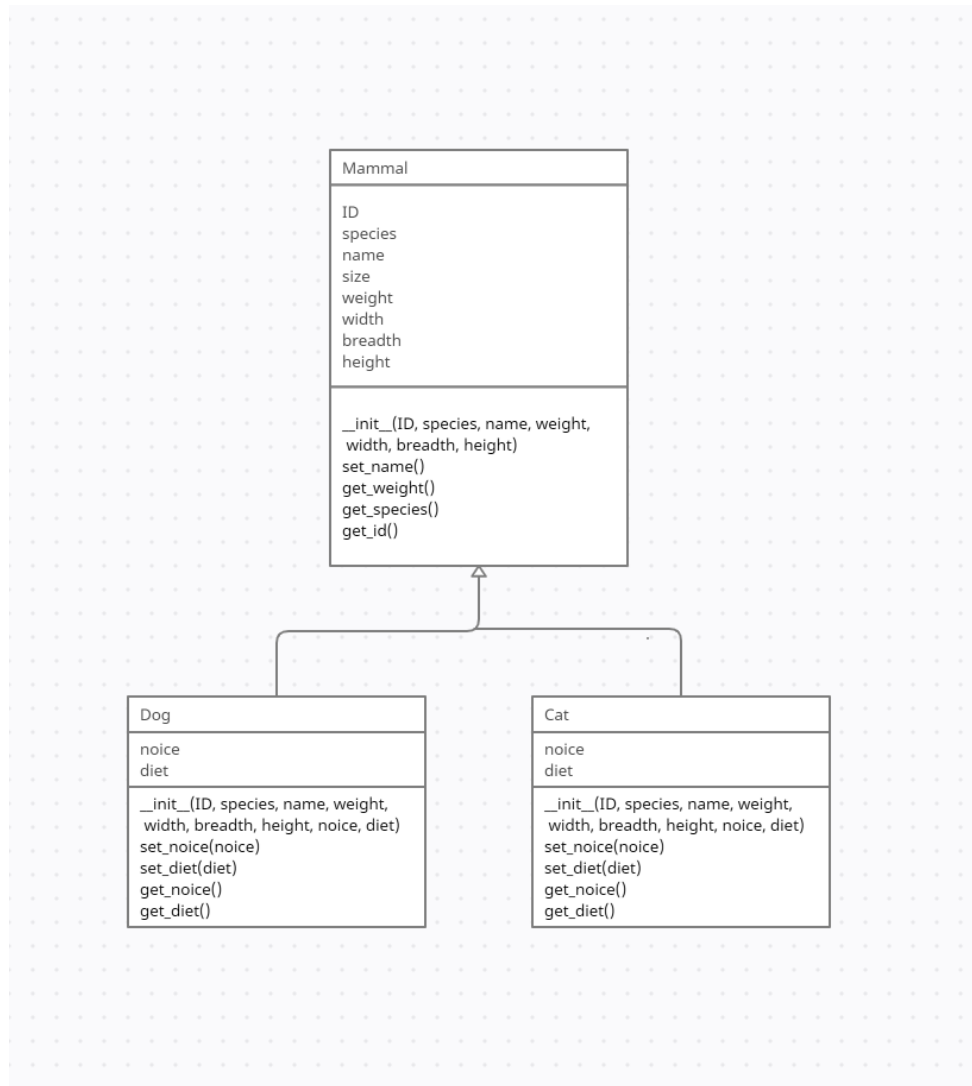
Platypus **is a** Wild_animal. Wild_animal **is a** Mammal

Student **is a** participant of OOP course.

Teacher **is a** participant of OOP course.

Jos tämä ei käy järkeen luokissa on jonkinlainen logiikka virhe.

2. Draw a UML diagram of task 3. Use UML syntax and see how data attributes and methods are presented in the example presented on class (about Automobile), can be found on lecture slides. You can use any software or e.g. draw by hand and take a picture. Example in lectures is drawn using MS Visio.



3. Inherit some animals from the Mammal class (that you created in Exercise 4). Add data attribute for the noise the animal makes and the diet they have. Display your objects on screen (= Print out the state of each object (use str-method)).

Screen capture of Task 3

Mammal class:

```
main.py x mammal_class.py x
1  # File name: mammal_class
2  # Author: Pekka Lehtola
3  # Description: class for creating mammals
4
5  class Mammal:
6
7      def __init__(self):
8
9          self.id = "ID"
10         self.species = "species"
11         self.name = "name"
12         self.weight = "weight"
13         self.width = 1
14         self.breadth = 1
15         self.height = 1
16         self.size = float((self.width * self.breadth * self.height) / (1000 * 1000)) #Calculates cm^3 and converts it to m^3
17
18         #str method for clean output printing with correct units
19         def __str__(self):
20
21             return f"""
22             ID: {self.id}
23             Species: {self.species}
24             Name: {self.name}
25             Size {self.size} m^3
26             Weight {self.weight} Kg
27             Width {self.width} cm
28             Breadth {self.breadth} cm
29             Height {self.height} cm
30             """
31
32         def set_id(self):
33             self.id = int(input("Enter an id for the animal: "))
34         def set_species(self):
35             self.species = (input("Enter a species for the animal: "))
36         def set_name(self):
37             self.name = str(input("Enter a name for the animal: "))
38         def set_weight(self):
39             self.weight = int(input("Enter a weight for the animal: "))
40         def set_width(self):
41             self.width = int(input("Enter a width for the animal: "))
42             self.size = float((self.width * self.breadth * self.height) / (1000 * 1000))
43         def set_breadth(self):
44             self.breadth = int(input("Enter a breadth for the animal: "))
45             self.size = float((self.width * self.breadth * self.height) / (1000 * 1000))
46         def set_height(self):
47             self.height = int(input("Enter a height for the animal: "))
48             self.size = float((self.width * self.breadth * self.height) / (1000 * 1000))
49
50         def get_id(self):
51             return self.id
52         def get_species(self):
53             return self.species
54         def get_name(self):
55             return self.name
56         def get_weight(self):
57             return self.weight
58         def get_width(self):
59             return self.width
60         def get_breadth(self):
61             return self.breadth
62         def get_height(self):
63             return self.height
```

Main:

```
main.py × mammal_class.py ×
1  # File name: main
2  # Author: Pekka Lehtola
3  # Description: Main function for exercise 6_3
4
5  from mammal_class import Mammal
6
7  #Dog inherited from Mammal class
8  class Dog(Mammal):
9
10     #Two new attributes noice and diet
11     #Rest are inherited
12     def __init__(self):
13         Mammal.__init__(self)
14         self.noice = "noice"
15         self.diet = "diet"
16
17     #Set and Get methods
18     def set_noice(self):
19         self.noice = str(input("Enter a noice for the animal: "))
20     def set_diet(self):
21         self.diet = str(input("Enter a diet for the animal: "))
22
23     def get_noice(self):
24         return self.noice
25     def get_diet(self):
26         return self.diet
27
28     #Str function for clear output prints.
29     def __str__(self):
30
31         return f"""
32         ID: {self.id}
33         Species: {self.species}
34         Name: {self.name}
35         Size: {self.size} m^3
36         Weight: {self.weight} Kg
37         Width: {self.width} cm
38         Breadth: {self.breadth} cm
39         Height: {self.height} cm
40         Noice: {self.noice}
41         Diet is {self.diet}
42         """
43
44     #Setting dog as object and giving it values.
45     dog = Dog()
46
47     dog.set_id()
48     dog.set_species()
49     dog.set_name()
50     dog.set_weight()
51     dog.set_width()
52     dog.set_breadth()
53     dog.set_height()
54     dog.set_diet()
55     dog.set_noice()
56
57     print(dog)
```

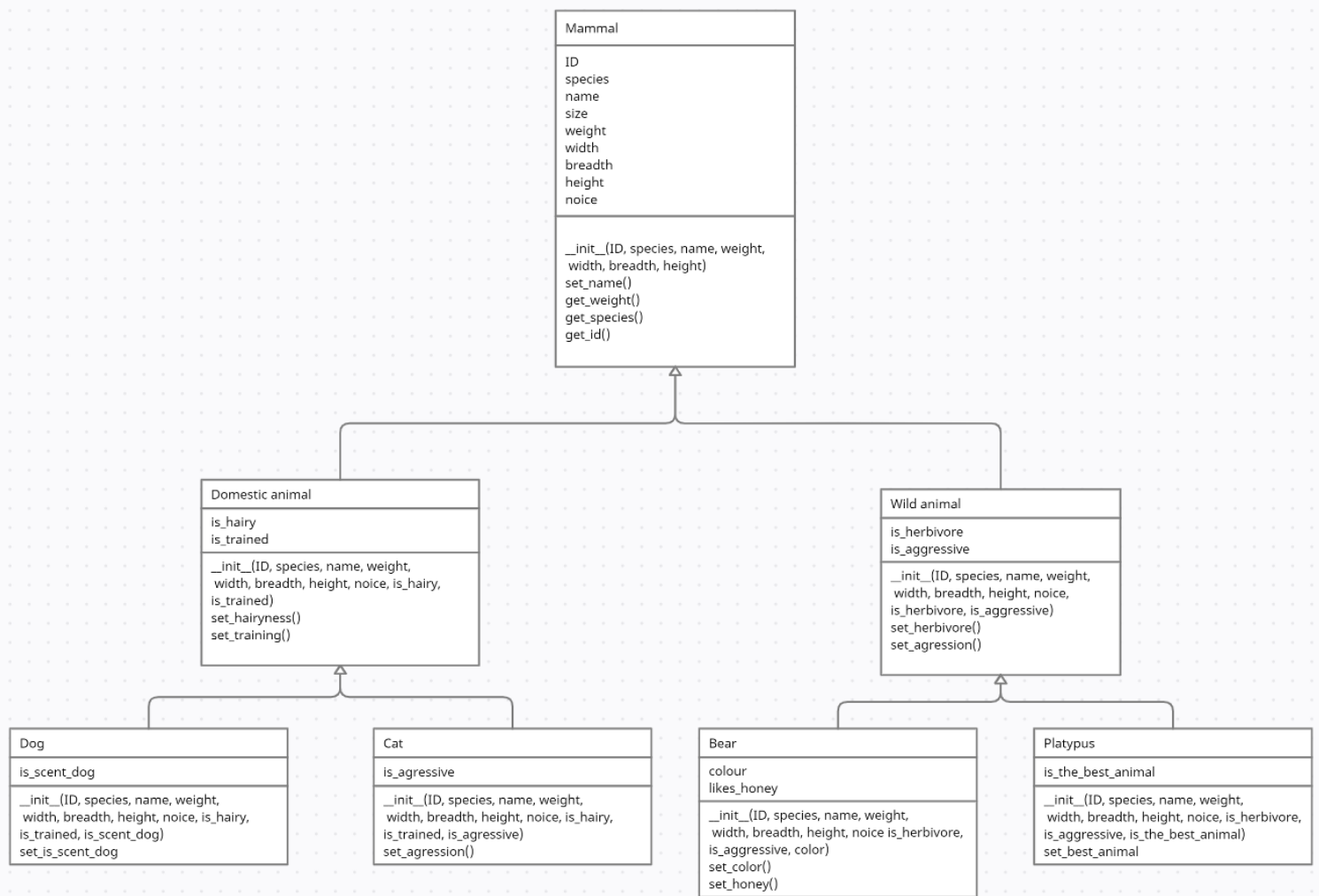
Screen capture of the output of Task 3

```
C:\Users\pekka\AppData\Local\Microsoft\WindowsApps\python3.7.exe "C:/Users/pekka/OneDrive -
Enter an id for the animal: 1
Enter a species for the animal: Dog
Enter a name for the animal: Mauri
Enter a weight for the animal: 66
Enter a width for the animal: 32
Enter a breadth for the animal: 50
Enter a height for the animal: 50
Enter a diet for the animal: 300g dog food twice a day
Enter a noise for the animal: Bark

ID: 1
Species: Dog
Name: Mauri
Size: 0.08 m^3
Weight: 66 Kg
Width: 32 cm
Breadth: 50 cm
Height: 50 cm
Noise: Bark
Diet is 300g dog food twice a day

Process finished with exit code 0
```

4. Draw a UML diagram of exercise 5. Use UML syntax and see how data attributes and methods are presented in the example presented on class (about Automobile etc.).



5. Change your task 3 code like this: Inherit a domestic animal from Mammal. Also inherit a wild animal from Mammal. Then inherit a few domestic and wild animals from those classes and print them out. Each mammal should make unique noise and have a certain diet as additional data attributes. Add some relevant attributes. Display your objects on screen.

Screen capture of Task 5

Mammal_class:

```
1  # File name: mammal_class
2  # Author: Pekka Lehtola
3  # Description: class for creating mammals
4
5  class Mammal:
6
7      def __init__(self):
8
9          self.id = "ID"
10         self.species = "species"
11         self.name = "name"
12         self.weight = "weight"
13         self.width = 1
14         self.breadth = 1
15         self.height = 1
16         self.size = float((self.width * self.breadth * self.height) / (1000 * 1000)) #Calculates cm^3 and converts it to m^3
17         self.noice = ""
18         self.diet = ""
19
20     #str method for clean output printing with correct units
21     def __str__(self):
22
23         return f"""
24         ID: {self.id}
25         Species: {self.species}
26         Name: {self.name}
27         Size: {self.size} m^3
28         Weight: {self.weight} Kg
29         Width: {self.width} cm
30         Breadth: {self.breadth} cm
31         Height: {self.height} cm
32         Noice: {self.noice}
33         Diet is {self.diet}
34         """
35
36     def set_id(self, id):
37         self.id = int(id)
38     def set_species(self, species):
39         self.species = species
40     def set_name(self, name):
41         self.name = name
42     def set_weight(self, weight):
43         self.weight = int(weight)
44     def set_width(self, width):
45         self.width = int(width)
46         self.size = float((self.width * self.breadth * self.height) / (1000 * 1000))
47     def set_breadth(self, breadth):
48         self.breadth = int(breadth)
49         self.size = float((self.width * self.breadth * self.height) / (1000 * 1000))
50     def set_height(self, height):
51         self.height = int(height)
52         self.size = float((self.width * self.breadth * self.height) / (1000 * 1000))
53     def set_noice(self, noice):
54         self.noice = noice
55     def set_diet(self, diet):
56         self.diet = diet
57
58     def get_id(self):
59         return self.id
60     def get_species(self):
61         return self.species
62     def get_name(self):
63         return self.name
64     def get_weight(self):
65         return self.weight
66     def get_width(self):
67         return self.width
68     def get_breadth(self):
69         return self.breadth
70     def get_height(self):
71         return self.height
72     def get_noice(self):
73         return self.noice
74     def get_diet(self):
75         return self.diet
76
```

Animals:

```
main.py × domestic_animals.py × wild_animals.py × animals.py × mammal_class.py ×
1  # File name: animals
2  # Author: Pekka Lehtola
3  # Description: Classes Wild_animal and Domestic_animal Inherited from Mammal class.
4
5  from mammal_class import *
6
7  # Inherits Wild animals from Mammal class.
8  class Wild_animal(Mammal):
9
10     # Wild animals have additional attributes Herbivore and Agression.
11     def __init__(self):
12         Mammal.__init__(self)
13         self.is_herbivore = True
14         self.is_agressive = False
15
16     def set_herbivore(self):
17
18         if self.is_herbivore:
19             self.is_herbivore = False
20         else:
21             self.is_herbivore = True
22
23     def set_agression(self):
24
25         if self.is_agressive:
26             self.is_agressive = False
27         else:
28             self.is_agressive = True
29
30     # Inherits Domestic_animal from Mammal class.
31     class Domestic_animal(Mammal):
32
33         # Domestic animals have additional attributes Hairyness and Training.
34         def __init__(self):
35             Mammal.__init__(self)
36             self.is_hairy = True
37             self.is_trained = True
38
39         def set_hairyness(self):
40
41             if self.is_hairy:
42                 self.is_hairy = False
43             else:
44                 self.is_hairy = True
45
46         def set_training(self):
47
48             if self.is_trained:
49                 self.is_trained = False
50             else:
51                 self.is_trained = True
```


Wild_animals:

main.py × domestic_animals.py × wild_animals.py × animals.py × mammal_class.py ×

```
1 # File name: wild_animals.
2 # Author: Pekka Lehtola
3 # Description: Bear and Platypus classes inherited from Wild_animals class.
```

```
4 from animals import *
```

```
5
6 # Inherits Bear from Wild_animals class
```

```
7 class Bear(Wild_animal):
```

```
8     def __init__(self):
```

```
9
10
11         Wild_animal.__init__(self)
12         self.colour = "Brown"
13         self.likes_honey = True
```

```
14
15 #Additional Attribute colour for bear class
```

```
16 def set_colour(self, colour):
17     self.colour = colour
```

```
18
19 # Calling this function changes the boolean value.
```

```
20 def set_likes_honey(self):
```

```
21
22     if self.likes_honey:
23         self.likes_honey = False
24     else:
25         self.likes_honey = True
```

```
26
27 #Bear class printing
```

```
28 def __str__(self):
```

```
29
30     return f"""
31     ID: {self.id}
32     Species: {self.species}
33     Name: {self.name}
34     Size: {self.size} m^3
35     Weight: {self.weight} Kg
36     Width: {self.width} cm
37     Breadth: {self.breadth} cm
38     Height: {self.height} cm
39     Noise: {self.noise}
40     Diet is {self.diet}
41     is herbivore: {self.is_herbivore}
42     is aggressive: {self.is_aggressive}
43     Colour is: {self.colour}
44     Likes honey: {self.likes_honey}
45     """
```

```
46
47 #Inherits Platypus from Wild animal class
```

```
48 class Platypus(Wild_animal):
```

```
49     def __init__(self):
```

```
50
51         Wild_animal.__init__(self)
52         self.is_the_best_animal = True
```

```
53
54 # Calling this function does nothing because platypus is the Superior animal.
```

```
55 def set_is_the_best_animal(self):
```

```
56
57     if self.is_the_best_animal:
58         self.is_the_best_animal = True
59     else:
60         self.is_the_best_animal = True
```

```
61
62 #Printing for platypus class.
```

```
63 def __str__(self):
```

```
64
65     return f"""
66     ID: {self.id}
67     Species: {self.species}
68     Name: {self.name}
69     Size: {self.size} m^3
70     Weight: {self.weight} Kg
71     Width: {self.width} cm
72     Breadth: {self.breadth} cm
73     Height: {self.height} cm
74     Noise: {self.noise}
75     Diet is {self.diet}
76     is herbivore: {self.is_herbivore}
77     is aggressive: {self.is_aggressive}
78     Is the best animal: {self.is_the_best_animal}
79     """
```

Domestic_animals:

```
main.py x domestic_animals.py x wild_animals.py x animals.py x mammal_class.py x
1  # File name: domestic_animals
2  # Author: Pekka Lehtola
3  # Description: Dog and Cat classes inherited from Domestic_animals class.
4
5  from animals import *
6
7  # Inherits Dog from Domestic animals.
8  class Dog(Domestic_animal):
9
10     def __init__(self):
11
12         Domestic_animal.__init__(self)
13         self.is_scent_dog = True
14
15     # Calling this function changes the boolean value.
16     def set_is_scent(self):
17
18         if self.is_scent_dog:
19             self.is_scent_dog = False
20         else:
21             self.is_scent_dog = True
22
23     # Defining Dog printing.
24     def __str__(self):
25
26         return f"""
27         ID: {self.id}
28         Species: {self.species}
29         Name: {self.name}
30         Size: {self.size} m^3
31         Weight: {self.weight} Kg
32         Width: {self.width} cm
33         Breadth: {self.breadth} cm
34         Height: {self.height} cm
35         Noice: {self.noice}
36         Diet is: {self.diet}
37         Is hairy: {self.is_hairy}
38         Is trained: {self.is_trained}
39         Is scent dog: {self.is_scent_dog}
40         """
41
42     # Inherits Cat from Domestic animals.
43     class Cat(Domestic_animal):
44
45         def __init__(self):
46             Domestic_animal.__init__(self)
47             self.is_agressive = True
48
49         # Calling this function changes the boolean value.
50         def set_agression(self):
51
52             if self.is_agressive:
53                 self.is_agressive = False
54             else:
55                 self.is_agressive = True
56
57         #Defining Cat printing.
58         def __str__(self):
59
60             return f"""
61             ID: {self.id}
62             Species: {self.species}
63             Name: {self.name}
64             Size: {self.size} m^3
65             Weight: {self.weight} Kg
66             Width: {self.width} cm
67             Breadth: {self.breadth} cm
68             Height: {self.height} cm
69             Noice: {self.noice}
70             Diet is: {self.diet}
71             Is_hairy: {self.is_hairy}
72             Is trained: {self.is_trained}
73             Is agressive: {self.is_agressive}
74             """
75
```

Main part 1:

```

1  # File name: main
2  # Author: Pekka Lehtola
3  # Description: Main function for exercise 6_5
4
5  #Importing Domestic animal and Wild animal classes.
6  from domestic_animals import *
7  from wild_animals import *
8
9  #Creates dog object with given attributes.
10 def dog_creating():
11
12     dog = Dog()
13
14     dog.set_id(1)
15     dog.set_species("Dog")
16     dog.set_name("Martti")
17     dog.set_weight(60)
18     dog.set_width(40)
19     dog.set_breadth(70)
20     dog.set_height(55)
21     dog.set_noice("BArk")
22     dog.set_diet("300g dog food twice a day.")
23     dog.set_is_scent()
24
25     print(dog)
26
27 #Creates cat object with given attributes.
28 def cat_creating():
29
30     cat = Cat()
31
32     cat.set_id(2)
33     cat.set_species("Cat")
34     cat.set_name("Liisa")
35     cat.set_weight(13)
36     cat.set_width(10)
37     cat.set_breadth(26)
38     cat.set_height(24)
39     cat.set_noice("Meow")
40     cat.set_training()
41     cat.set_diet("One can of cat food three time a day")
42
43     print(cat)
44

```

Main part 2:

```

44
45 #Creates bear object with given attributes.
46 def bear_creation():
47
48     bear = Bear()
49
50     bear.set_id(3)
51     bear.set_species("Bear")
52     bear.set_name("Pooh")
53     bear.set_weight(700)
54     bear.set_width(80)
55     bear.set_breadth(130)
56     bear.set_height(99)
57     bear.set_noice("Roar")
58     bear.set_diet("One moose a day")
59     bear.set_herbivore()
60     bear.set_agression()
61     bear.set_colour("Black")
62     bear.set_likes_honey()
63
64     print(bear)
65
66 #Creates platypus object with given attributes.
67 def platypus_creation():
68
69     platypus = Platypus()
70
71     platypus.set_id(4)
72     platypus.set_species("Platypus")
73     platypus.set_name("Perry")
74     platypus.set_weight(30)
75     platypus.set_width(14)
76     platypus.set_breadth(60)
77     platypus.set_height(26)
78     platypus.set_noice("GRRrrr")
79     platypus.set_diet("Plenty of lillypads")
80
81     print(platypus)
82
83
84
85 dog_creating()
86 cat_creating()
87 bear_creation()
88 platypus_creation()

```

Screen capture of the output of Task 5

```
main (3) x
C:\Users\pekka\AppData\Local\Microsoft\WindowsApps\python3.7.exe "C:/Users/pekka/OneDrive - Turun ammattikorkeakoulu,

ID: 1
Species: Dog
Name: Martti
Size: 0.154 m^3
Weight: 60 Kg
Width: 40 cm
Breadth: 70 cm
Height: 55 cm
Noise: Bark
Diet is: 300g dog food twice a day.
Is hairy: True
Is trained: True
Is scent dog: False

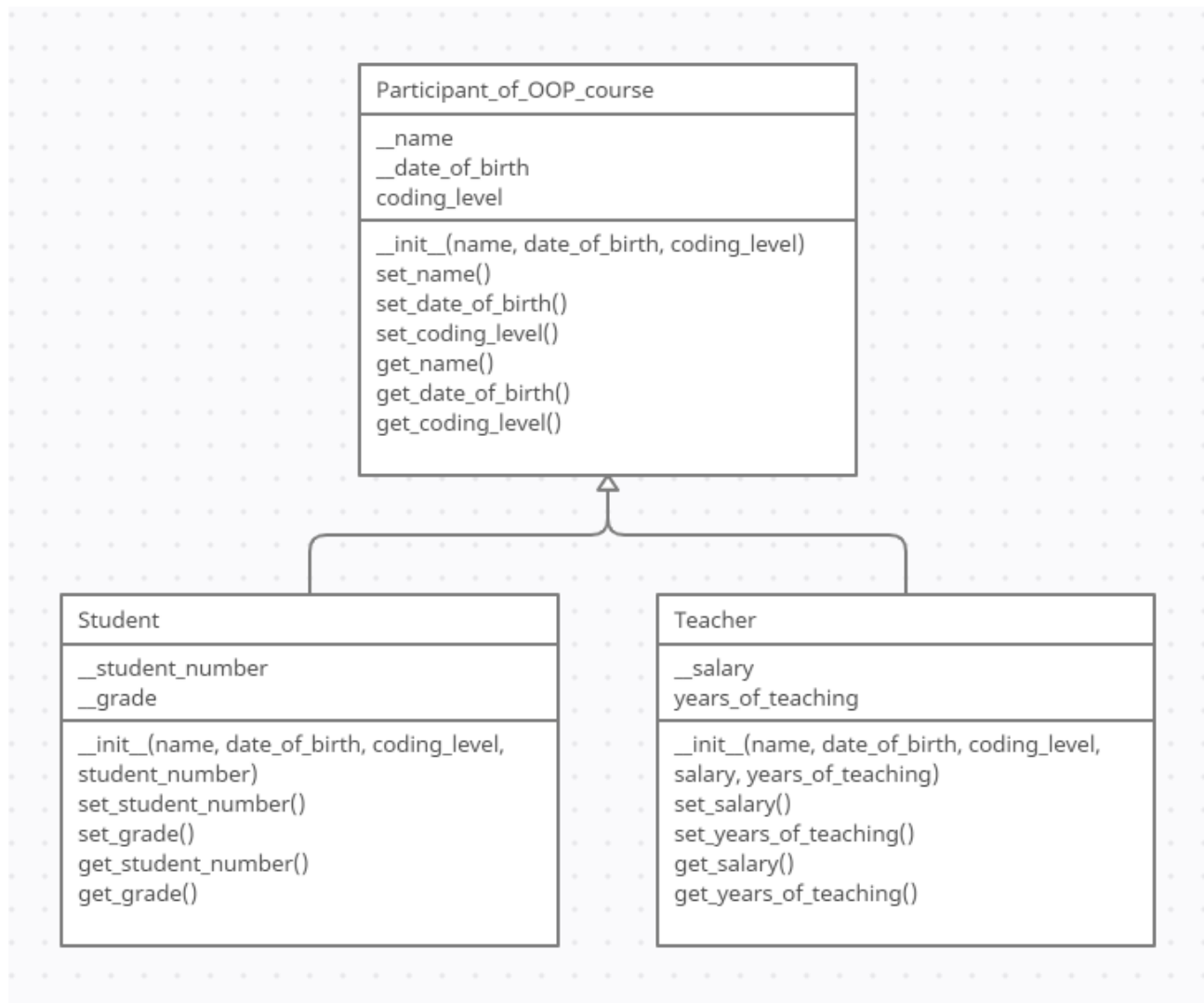
ID: 2
Species: Cat
Name: Liisa
Size: 0.00624 m^3
Weight: 13 Kg
Width: 10 cm
Breadth: 26 cm
Height: 24 cm
Noise: Meow
Diet is: One can of cat food three time a day
Is_hairy: True
Is trained: False
Is aggressive: True

ID: 3
Species: Bear
Name: Pooh
Size: 1.0296 m^3
Weight: 700 Kg
Width: 80 cm
Breadth: 130 cm
Height: 99 cm
Noise: Roar
Diet is One moose a day
is herbivore: False
is aggressive: True
Colour is: Black
Likes honey: False

ID: 4
Species: Platypus
Name: Perry
Size: 0.02184 m^3
Weight: 30 Kg
Width: 14 cm
Breadth: 60 cm
Height: 26 cm
Noise: GRRrr
Diet is Plenty of lillypads
is herbivore: True
is aggressive: False
Is the best animal: True

Process finished with exit code 0
```

6. Draw a UML diagram of tasks 7. Use UML syntax and see how data attributes and methods are presented in the example presented on class (about Automobile etc.).



7. Inherit a student and teacher from a participant of OOP course. Think a few proper data attributes that are 1) common for both teachers and students and 2) different between teachers and students.

Screen capture of Task 7

Participant_of_OOP_course:

```
main.py × teacher.py × student.py × participant_of_OOP_course.py ×
Q- Cc W .* 0 results ↑ ↓
1 # File name: participants_of_OOP_course
2 # Author: Pekka Lehtola
3 # Description: OOP_course class with common attributes.
4
5 #OOP_course_class
6 class OOP_course:
7
8     #Common attributes in course name, date of birth and coding level.
9     def __init__(self):
10
11         self.__name = "Jokunen"
12         self.__date_of_birth = "01012000"
13         self.coding_level = 1
14
15     def set_name(self, name):
16         self.__name = name
17
18     def set_date_of_birth(self, date_of_birth):
19         self.__date_of_birth = date_of_birth
20
21     def set_coding_level(self, coding_level):
22         self.coding_level = coding_level
23
24     def get_name(self):
25         return self.__name
26
27     def get_date_of_birth(self):
28         return self.__date_of_birth
29
30     def get_coding_level(self):
31         return self.coding_level
32
33     def __str__(self):
34
35         return f"""
36         Name: {self.__name}
37         Date of birth: {self.__date_of_birth}
38         Coding_level: {self.coding_level} """
```

Student:

```
main.py x teacher.py x student.py x participant_of_OOP_course.py x
# File name: student
# Author: Pekka Lehtola
# Description: Student class inherited from OOP_course class

from participant_of_OOP_course import *

#Student class inherited from OOP_course
class Student(OOP_course):
    #Student has additional attributes student number and grade

    def __init__(self):
        OOP_course.__init__(self)
        self.__student_number = "000000"
        self.__grade = "N/A"

    def set_student_number(self, student_number):
        self.__student_number = student_number

    def set_grade(self, grade):
        self.__grade = grade

    def get_student_number(self):
        return self.__student_number

    def get_grade(self):
        return self.__grade

    def __str__(self):
        return f"""
        Name: {self.get_name()}
        Date of birth: {self.get_date_of_birth()}
        Coding_level: {self.coding_level}
        Student number: {self.get_student_number()}
        Grade: {self.get_grade()} """
```

Teacher:

```
main.py x teacher.py x student.py x participant_of_OOP_course.py x
# File name: teacher
# Author: Pekka Lehtola
# Description: Teacher class inherited from OOP_course class

from participant_of_OOP_course import *

#Teacher class inherited from OOP_course
class Teacher(OOP_course):
    #Teacher has additional attributes salary and years of teaching.

    def __init__(self):
        OOP_course.__init__(self)
        self.__salary = 0
        self.years_of_teaching = 0

    def set_salary(self, salary):
        self.__salary = salary

    def set_years_of_teaching(self, years_of_teaching):
        self.years_of_teaching = years_of_teaching

    def get_salary(self):
        return self.__salary

    def get_years_of_teaching(self):
        return self.years_of_teaching

    def __str__(self):
        return f"""
        Name: {self.get_name()}
        Date of birth: {self.get_date_of_birth()}
        Coding_level: {self.coding_level}
        Salary: {self.get_salary()}
        Years of teaching: {self.years_of_teaching} """
```

Main:

```
main.py x teacher.py x student.py x participant_of_OOP_course.py x
1  # File name: main
2  # Author: Pekka Lehtola
3  # Description: Main file for exercise6_7
4
5  from student import *
6  from teacher import *
7
8  #Creates student object from Student class
9  def student_creation():
10
11     pekka = Student()
12
13     #Attribute values are set here..
14     pekka.set_name("Pekka")
15     pekka.set_date_of_birth("04021998")
16     pekka.set_coding_level("Good")
17     pekka.set_student_number("199321")
18     pekka.set_grade(5)
19
20     #Object printing
21     print(pekka)
22
23     #Creates Teacher object from Teacher class
24     def teacher_creation():
25
26         sanna = Teacher()
27
28         sanna.set_name("Sanna")
29         sanna.set_date_of_birth("03031990")
30         sanna.set_coding_level("Exelent")
31         sanna.set_salary(3200)
32         sanna.set_years_of_teaching(8)
33
34         print(sanna)
35
36     #Function activation happens here..
37     student_creation()
38     teacher_creation()
```

Screen capture of the output of Task 7

```
main (4) x
C:\Users\pekka\AppData\Local\Microsoft\Wind

Name: Pekka
Date of birth: 04021998
Coding_level: Good
Student number: 199321
Grade: 5

Name: Sanna
Date of birth: 03031990
Coding_level: Exelent
Salary: 3200
Years of teaching: 8

Process finished with exit code 0
```


8. Each participant of Task 7 has also 1 domestic animal and 1 wild animal. Display the teachers, students and their information.

Files that are unchanged : animals, mammal_class, wild_animals, teacher, student, wild_animals, domestic_animals

Screen capture of Task 8

Small edits to animal creation functions (**Main in task 5**) and person creation functions (**Main in task 7**):

```
#Creates dog object with given attributes.
def dog_creation():

    global dog

    dog = Dog()

#Creates cat object with given attributes.
def cat_creation():

    global cat

    cat = Cat()

#Creates bear object with given attributes.
def bear_creation():

    global bear

    bear = Bear()

#Creates platypus object with given attributes.
def platypus_creation():

    global platypus

    platypus = Platypus()
```

```
def student_creation():

    global pekka

    pekka = Student()

#Creates Teacher object from Teacher class
def teacher_creation():

    global sanna

    sanna = Teacher()
```

Edits to OOP_course class (From task 7):

```
#OOP course class
class OOP_course:

    #Common attributes in course name, date of birth and coding level.
    def __init__(self):

        self.__name = "Jokunen"
        self.__date_of_birth = "01012000"
        self.coding_level = 1
        self.domestic_animal = None
        self.wild_animal = None
```

```
36
37     def set_domestic_animal(self, domestic):
38         self.domestic_animal = domestic
39
40     def set_wild_animal(self, wild):
41         self.wild_animal = wild
42
43     def get_domestic_animal(self):
44         return self.domestic_animal
45
46     def get_wild_animal(self):
47         return self.wild_animal
48
```

Main:

```
main.py × animal_creation.py × person_creation.py × participant_of_OOP_course.py ×
1  # File name: main
2  # Author: Pekka Lehtola
3  # Description: Main file for exercise6_8
4
5  #Importing function for creating each animal and creator for student and teacher..
6  from animal_creation import *
7  from person_creation import *
8
9  #Creates student pekka and teacher sanna that are global objects
10 student_creation()
11 teacher_creation()
12
13 #Creates each animal and set them as global objects
14 dog_creating()
15 cat_creating()
16 bear_creation()
17 platypus_creation()
18
19 #Sets up Pekkas animals
20 pekka.set_domestic_animal(dog)
21 pekka.set_wild_animal(bear)
22
23 #Sets up Sannas animals
24 sanna.set_domestic_animal(cat)
25 sanna.set_wild_animal(platypus)
26
27 #First prints Pekka object and then his animal attributes
28 print(pekka)
29 print("    Pekka has the following animals: ")
30 print(pekka.get_domestic_animal())
31 print(pekka.get_wild_animal())
32
33 #First prints Sanna object and then her animals..
34 print(sanna)
35 print("    Sanna has the following animals: ")
36 print(sanna.get_domestic_animal())
37 print(sanna.get_wild_animal())
```

Error highlights are caused from computer not recognizing global objects.

I didn't add all the code here because document would have been way too long.

Screen capture of the output of Task 8

Output part 1:

```
main (5) ×
C:\Users\pekka\AppData\Local\Microsoft\WindowsApps\

Name: Pekka
Date of birth: 04021998
Coding_level: Good
Student number: 199321
Grade: 5

Pekka has the following animals:

ID: 1
Species: Dog
Name: Martti
Size: 0.154 m^3
Weight: 60 Kg
Width: 40 cm
Breadth: 70 cm
Height: 55 cm
Noice: BArk
Diet is: 300g dog food twice a day.
Is hairy: True
Is trained: True
Is scent dog: False

ID: 3
Species: Bear
Name: Pooh
Size: 1.0296 m^3
Weight: 700 Kg
Width: 80 cm
Breadth: 130 cm
Height: 99 cm
Noice: Roar
Diet is One moose a day
is herbivore: False
is aggressive: True
Colour is: Black
Likes honey: False
```

Output part 2:

```
Name: Sanna
Date of birth: 03031990
Coding_level: Exelent
Salary: 3200
Years of teaching: 8

Sanna has the following animals:

ID: 2
Species: Cat
Name: Liisa
Size: 0.00624 m^3
Weight: 13 Kg
Width: 10 cm
Breadth: 26 cm
Height: 24 cm
Noice: Meow
Diet is: One can of cat food three time a day
Is_hairy: True
Is trained: False
Is aggressive: True

ID: 4
Species: Platypus
Name: Perry
Size: 0.02184 m^3
Weight: 30 Kg
Width: 14 cm
Breadth: 60 cm
Height: 26 cm
Noice: GRRrr
Diet is Plenty of lillypads
is herbivore: True
is aggressive: False
Is the best animal: True

Process finished with exit code 0
```

Screen capture of git log (showing that you made a commit after every task).

🔍 master ▾ Object_Oriented_Programming / OOP / Exercise6 / Go to file

| | | | |
|--------------------------------|-----------------------|-----------------------|---------------|
| 🔗 pele98 Exercise6_8 version 1 | | 25a5fc8 3 minutes ago | 🕒 History |
| .. | | | |
| 📁 Exercise6_3 | Exercise6_3 version 1 | | yesterday |
| 📁 Exercise6_5 | Exercise6_5 version 2 | | 2 hours ago |
| 📁 Exercise6_8 | Exercise6_8 version 1 | | 3 minutes ago |
| 📁 exercise6_7 | Exercise6_7 version 1 | | 1 hour ago |
| 📄 Exercise6.pdf | Exercise6_3 version 1 | | yesterday |

Self-assessment:

This exercise was easy/difficult/ok/etc. for me because...

Nämä tehtävä koin melko helpoiksi, haastetta kyllä toi se kun jaoin luokat ja funktiot eri tiedostoihin ja sen kautta tuli muutama ongelma.

Doing this exercise, I learned...

Periytyvyyttä ja sen että objectin attribuutti voi olla myös kokonainen objecti.

Oppisin myös käyttämään UML charttia jonka huomasin olevan paljon kätevämpi koodin hahmottelussa, kuin pseudo koodi.

I am still wondering...

Voiko __str__ function periä myös ja siihen tehdä muokkauksia perityn luokan sisällä.

I understood/did not understand that... ; I did/did not know that... ; I did/did not manage to do...

-